DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

EE491 PROJECT REPORT

PRINTMANAGER
An Automated Configuration and Management System for Printers with Raspberry Pi

Student Name: Ecem Naz Fidan

Student ID: 280206048

Advisor: Prof. Dr. Bilge Karaçalı

DATE: 15/02/2025

ABSTRACT

In today's world, a device needs network accessibility, so Wi-Fi connectivity is one of the most essential features of evolving printer technology. However, this has created the need to replace older devices that do not have this feature. This project aims to build a plug-and-play automatization device using Raspberry Pi and the CUPS (Common UNIX Printing System) framework that makes USB-connected printers without a Wi-Fi connectivity network accessible. The primary goal is to eliminate the difficulties associated with configuring and operating various printers, especially older models with limited driver support, by streamlining the detection, installation, and configuration processes.

Since printers have specific protocols and driver systems, this remains an ongoing challenge, especially when trying to integrate older models into new systems. Typical methods require manual driver installation and troubleshooting, which can be time-consuming and error-prone. Although there have been partial solutions, such as using Wi-Fi dongles or adapters to make printers wireless, vendor-specific software or cloud printing services, virtual machines to emulate legacy environments, etc., no unified and automated approach has been implemented to address the various printer types effectively.

This project develops a device that automates the detection of printers and their integration into the Common UNIX Printing System (CUPS). Also, wireless printing is enabled by creating an access point and providing AirPrint protocol support. The first feature starts by extracting the identifiers of USB-connected printers, detecting their manufacturer and model, and allowing them to be registered seamlessly with the CUPS server in a compatible and standardized format. The old printer's driver can then be located and ready for use via CUPS. The second functionality enables wireless printing via the AirPrint protocol for devices on the local network created by setting up a custom access point on the Raspberry Pi and using the Avahi daemon service. All the mentioned operations are structured using bash scripts with system configurations such as udev rules, systemd service files, and cron jobs that ensure their organized functioning. The resulting device simplifies the complex and manual printing process by seamlessly integrating printers into the network and enabling AirPrint to be controlled from modern devices. The resulting device simplifies the complex and manual printing process by seamlessly integrating printers into the network and enabling AirPrint to be controlled from modern devices. The project bridges the compatibility gap between legacy printer models and current networked environments while demonstrating the ability of open-source tools to provide a practical solution.

TABLE OF CONTENTS

ABBREVIATIONS

CUPS: Common UNIX Printing System

IPP: Internet Printing Protocol

mDNS: Multicast Domain Name System

Avahi: (Specific to Avahi project, not expanded further in the document)

Wi-Fi: Wireless Fidelity

USB: Universal Serial Bus

POSIX: Portable Operating System Interface

URI: Uniform Resource Identifier

LCD: Liquid Crystal Display

DNS: Domain Name System

IEEE: Institute of Electrical and Electronics Engineers

LIST OF FIGURES

LIST OF TABLES

This report doesn't contain any tables.

1. INTRODUCTION

The rapid evolution of networked environments has made Wi-Fi connectivity an essential feature for modern printers, while only USB-connected printers are difficult to integrate into modern systems. These printers lack network accessibility and driver support, increasing the need for solutions to address this technological gap. Literature has emphasized that manual driver installation and configuration methods are generally required to make printers usable, leading to inefficiencies when working with older devices.

Research into improving the network accessibility of printers has examined a variety of approaches. Wi-Fi adapters and dongles have been used to upgrade older printers. Even though limited support for specific models, has provided them with network capabilities [1][2]. Similarly, cloud-based solutions such as Google Cloud Print aimed to simplify printer access by abstracting connectivity constraints however, they posed limitations to internet availability and long-term viability [3]. The Common UNIX Printing System (CUPS), an open-source platform, and Avahi, a network protocol, have been effective solutions for the dynamic discovery and management of printers. However, these tools have been underutilized due to their limited compatibility with legacy printers and the fact that the automation process can seem complex for users [4][5]. These tools have been used in this project effectively due to the flexibility and customization possibilities provided by their open-source nature.

This project provided a solution to these challenges by developing a plug-and-play device that leverages Raspberry Pi and open-source systems to automate printer management. The proposed solution implements the detection and integration of USB-connected printers into CUPS and the setup of a wireless access point to enable AirPrint functionality for seamless printing from local devices. The automated system, compatible with open-source tools, was built using Bash scripts.

1.1. Project Overview

It was aimed to solve the network integration problem of the printers described by creating a Raspberry Pi based device. In this project, Raspberry Pi was preferred because it offers both an economical solution and supports powerful open-source tools. Raspberry Pi provides a flexible platform for combining various operations such as printer detection, network integration and wireless printing on a compact device. The device first identifies the printers connected via USB, determines the manufacturer and model information, and registers the printers in a compatible format on CUPS. Thus, the device aims to save time for users by eliminating manual installation processes.

The second function of the project is to turn the Raspberry Pi into a hotspot and print wirelessly from a smartphone, etc. via AirPrint. This function facilitates the discovery of printers on the local network using the zero-configuration network policies of the Avahi-daemon service and ensures compatibility with modern devices.

Other methods proposed in the literature have significant disadvantages compared to the approach proposed in this project. For instance, Wi-Fi dongles or adapters have made older printers wireless, but with limited model support and dependency on vendor-specific software. Cloud-based services, such as Google Cloud Print, facilitated printer access over the internet, but they had shortcomings, including internet dependency, and were subsequently discontinued. Virtual machines were used to emulate legacy systems but were far from user-friendly due to complex configuration requirements. The common problem with these methods is that they have a high manual processing load and slow down the integration process considerably. Hence, they are not preferred solutions by users, and this has resulted in the replacement of existing printers with new generation printers.

2. PROBLEM DEFINITON

To adapt to today's needs, older printers need to adapt to modern network connectivity features such as Wi-Fi or AirPrint. Configuring these legacy printers often requires manual installation of drivers and manual configuration of devices. This process is time-consuming, and therefore disadvantageous. There are also factors that make it difficult to integrate printers into network systems, such as lack of driver support for older devices.

2.1. Key Issues

1. **Limited Connectivity**
   Older printers without network connectivity are wirelessly inaccessible. This disadvantage makes it difficult to incorporate these devices into modern networked printing solutions.

2. **Complex Configuration**

Integrating legacy printers into modern network systems poses a significant problem for users due to the complexity of manual installation processes. Even the availability of support for solutions offered by manufacturers is a major obstacle to the usability of many printers. Other complexities, such as finding appropriate drivers, resolving naming inconsistencies, and configuring devices in a network environment, also made access difficult for non-technical users. Other approaches we did not address in the project include external adapters to add wireless connectivity capabilities, internet-based access solutions, and virtual machines to emulate legacy systems. Nevertheless, these methods have disadvantages such as limited model support, dependency on internet connectivity, or complex configuration processes. Thus, existing solutions can't provide a full automation and integration process and do not ensure a user-friendly experience. [6]

3. **Cost and Environmental Impact**

As a result of the other problems identified, old printers are being replaced with new devices with network capabilities, even if those printers are still usable. This incurs high costs and increases electronic waste. In other words, there is a need for sustainable solutions that can extend the lifespan of old devices.

2.2. Problem Statement

The project aims to automate the integration of old printers into modern network systems and facilitate configuration processes. Using a Raspberry Pi as the central processing unit, this system enables USB-connected printers to be detected, their manufacturer and model information determined and automatically added to the CUPS server. This eliminates the necessity for manual intervention and speeds up compatibility processes.

Another important function offered by the system is wireless printing. A local network is established through a special hotspot generated on the Raspberry Pi and enables AirPrint functionality on this network and later on the general Wi-fi network. AirPrint functionality works along with the Avahi daemon service in order to enable other smart devices (e.g. smartphones and tablets) to automatically discover printers. Avahi is an implementation of the mDNS protocol that facilitates device discovery and service sharing. This framework ensures that devices are easily accessible and available on the network. [5]

By minimizing user intervention, this integrated solution is designed to automate printer integration processes and reduce dependency on external tools. It can also ensure the sustainable compatibility of legacy printers with modern systems.

3. PROPOSED SOLUTION

In this section, the methodological foundations and implementation processes of the solution developed within the scope of the project will be explained in detail. The developed system offers a user-friendly and sustainable environment for automating the integration of printers into network environments and providing wireless printing support. The design of the solution aims to provide automatic detection, configuration and management of printers via CUPS (Common UNIX Printing System) as well as wireless printing support via the AirPrint protocol using mDNS principles with the Avahi daemon service. Furthermore, it aims to provide both technical flexibility and broad device support by effectively combining open-source tools. The proposed device is given in Figure 1 and 2.

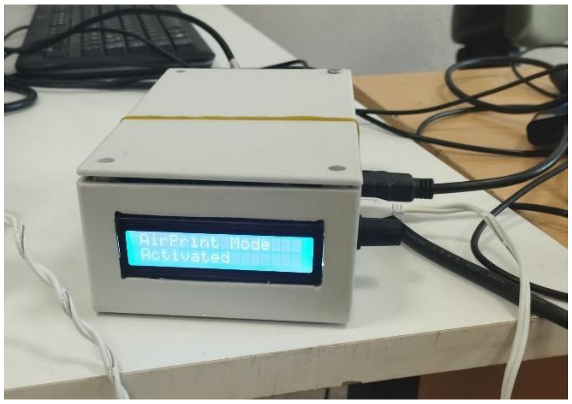


Figure 1: The image of PrintManager and its screen

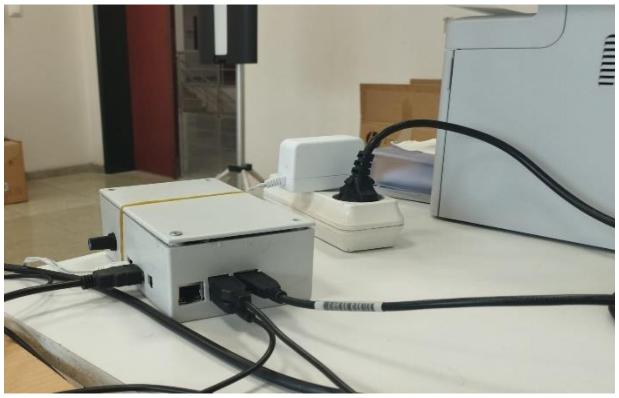


Figure 2: Image of PrintManager and USB Ports

3.1. System Design and Implementation

At the heart of the project is the Raspberry Pi device, with the aim of providing a suitable solution to modern networking requirements. The Raspberry Pi was chosen for this project because it is a low-cost, compact and easy to work on device. The device's Linux-based operating system enables it to work in compatibility with open source tools and offers seamless integration to modern systems as well as legacy devices.

The Raspberry Pi was used in the project to perform both the automatic detection and integration of printers into the CUPS (Common UNIX Printing System) server and to serve as a hotspot for wireless printing.

3.1.1 Hardware Configuration

The hardware infrastructure of the project consists of integrated components that are assembled together for the functionality of the solution. These components have been selected and configured to support the functions required for successful operation.

The hardware components used in the project can be listed as follows:

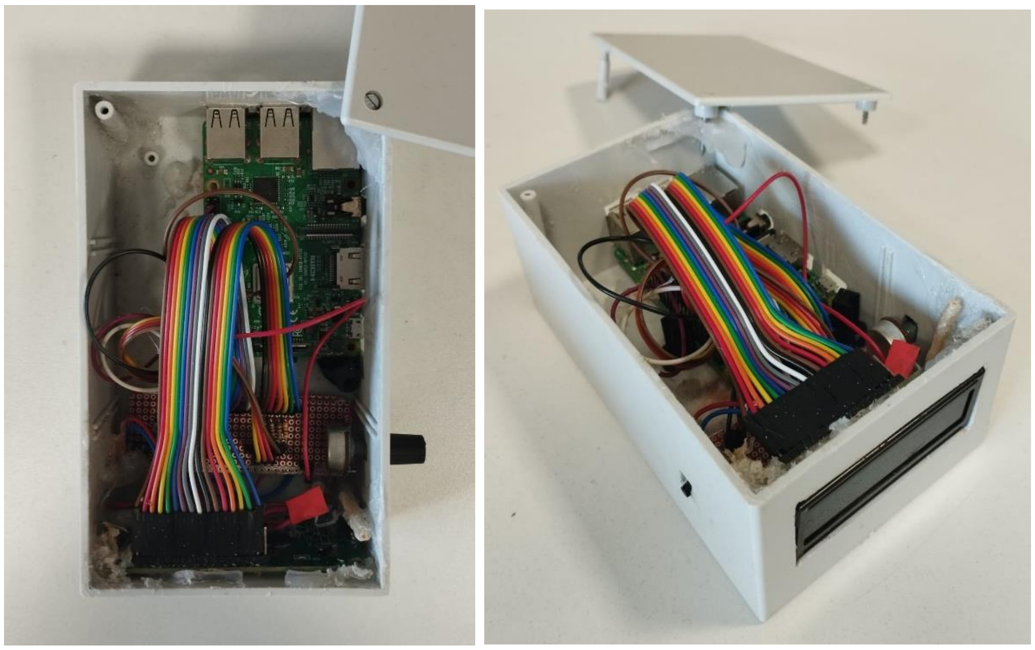
Raspberry Pi: It functions as the central processing unit of the project device. This component handles all basic functions such as printer detection, driver integration and network management.

LCD Display: Acts as the user feedback mechanism. Information such as the status of the device and the progress is visually presented to the users through this screen.

USB Ports: Allows printers to be physically connected and detected. The USB interface is an essential component for recognizing data protocols from the printer and selecting the appropriate driver.

AirPrint Button: Controls enabling and disabling wireless printing mode. This physical button increases the usability of the device and allows you to manually manage the mode switching.

The hardware components of the device are shown in Figure 3.



a) b)

Figure 3: The hardware of PrintManager

3.1.2 Software Stack

The software infrastructure of the project is composed of open-source tools and services to perform basic functions such as detecting printers, assigning drivers, providing network connectivity and wireless printing. These tools enable the system to work with both legacy and modern printers:

CUPS (Common UNIX Printing System):

CUPS is the core software service used for printer management. It detects printers both locally and over the network, configures them with the appropriate drivers, and manages printing processes. In the system, CUPS enables automatic registration of printers and queuing of print jobs. CUPS also enables printer integration by providing advanced features such as managing drivers and prioritizing print jobs. [4]

Avahi-Daemon:

Avahi-daemon is a service discovery tool based on mDNS (Multicast DNS) that detects printers by other devices on the same network. This has been used specifically for enabling the AirPrint protocol. Avahi makes the printer visible to devices on the local network and allows them to initiate wireless printing. [5]

Bash Scripts:

Custom bash scripts were created in the project to automate processes such as printer detection, assigning appropriate drivers and managing network settings. These scripts allow the system to run without the need for user intervention.

System Configuration Tools:

udev Rules: Automatically triggers the detection process when USB printers are connected and starts the script file that adds the printer to the CUPS server.
Systemd Services: Runs the script file that manages the startup and shutdown sequences of critical processes such as wireless access point creation and Avahi daemon.
Cron Tasks: Runs the script file that should start with the system when the system starts.

3.1.3 Automation and Maintenance

The automation mechanisms included in the project to automatically trigger the appropriate scripts in certain situations. These mechanisms include the following:

Cron Tasks: Used to initialize the list_printer.sh script that lists the available printers on the CUPS server at system startup.

Udev Rules: Detects hardware events such as USB printers being connected or removed and automatically triggers the relevant scripts. Therefore, it is possible to quickly detect and configure printers.

Systemd Services: Provides organized and seamless operation of AirPrint mode in the background to support wireless printing functionality. It is used for stable and uninterrupted operation of the system.

These automation processes have the task of triggering certain events and ensuring the stability of the system. The management processes were handled in detail in the scripts developed within the project. The operation of all system and script files is represented in the block diagram in Figure 4.

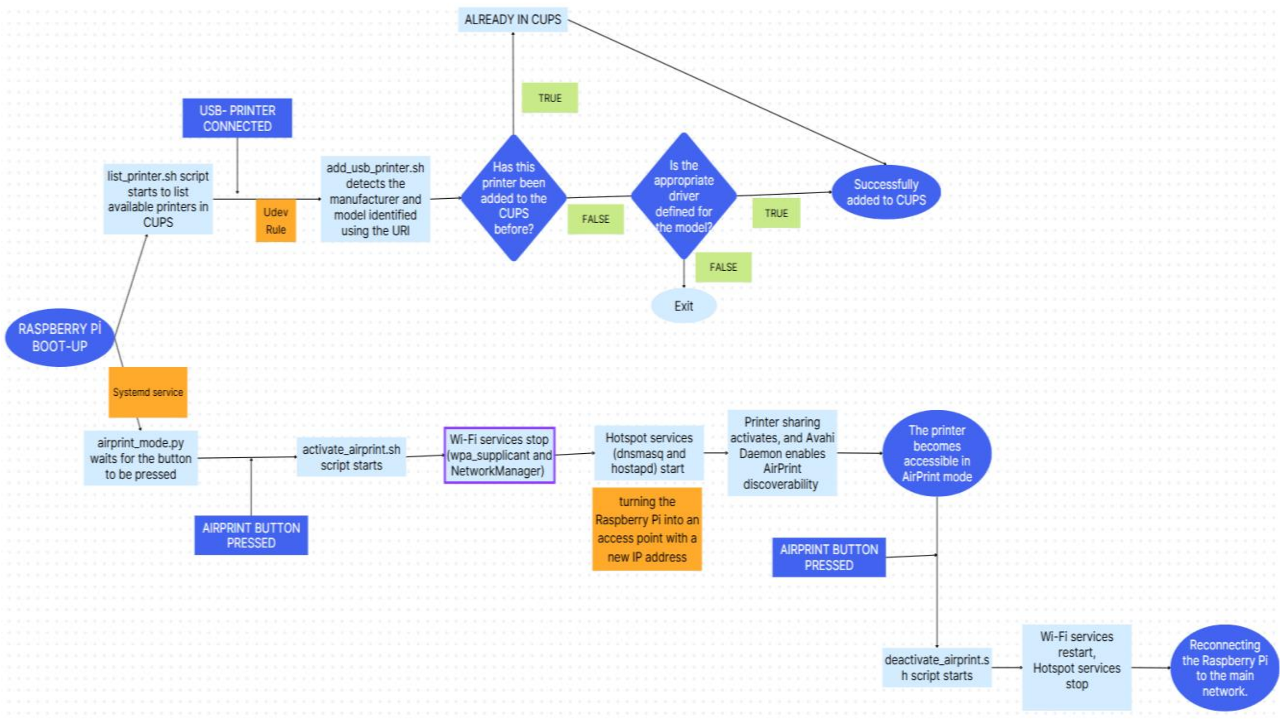

ALREADY IN CUPS
TRUE
USB- PRINTER CONNECTED
list_printer.sh script starts to list available printers in CUPS
Udev Rule
add_usb_printer.sh detects the manufacturer and model identified using the URI
Has this printer been added to the CUPS before?
FALSE
Is the appropriate driver defined for the model?
TRUE
FALSE
Exit
Successfully added to CUPS
RASPBERRY Pİ BOOT-UP
Systemd service
airprint_mode.py waits for the button to be pressed
AIRPRINT BUTTON PRESSED
activate_airprint.sh script starts
Wi-Fi services stop (wpa_supplicant and NetworkManager)
Hotspot services (dnsmasq and hostapd) start
turning the Raspberry Pi into an access point with a new IP address
Printer sharing activates, and Avahi Daemon enables AirPrint discoverability
The printer becomes accessible in AirPrint mode
AIRPRINT BUTTON PRESSED
deactivate_airprint.sh script starts
Wi-Fi services restart, Hotspot services stop
Reconnecting the Raspberry Pi to the main network.

Figure 4: Operation diagram of the Custom Scripts in the system

3.2. Review of Custom Scripts and Their Functions

The software infrastructure developed in the project was built with a series of custom scripts to ensure the proper functioning of automatic discovery, printer configuration and wireless printing functions. It is designed to increase the efficiency of the system and automate manual processes. The functions of each script and its significance in the project are explained in detail, and all the custom script files referenced in this report are provided in the Appendix section for further review.

3.2.1. list_printer.sh

The script is designed to list all the printers registered on the CUPS server. It runs automatically as a cron task at system startup to check the current status of the printers and prevent any instability in the system. The listed printer information is intended to guide the management and configuration processes.

3.2.2. add_usb_printer.sh

Once a USB printer is connected, the script is activated and detects the printer, then extracts the model and manufacturer information from the URI and adds it to the CUPS server. It integrates with Udev rules and is automatically triggered when the printer is connected. The script checks if the printer has already been added to the system and if a new printer is detected, it finds and configures the appropriate driver. If the correct driver compatible with CUPS is not found, it notifies "Driver not found download manually" via the lcd screen. Or if there is a naming error, it reports "Error adding to CUPS" and exits the script. Finally, the script successfully integrates old printers into modern systems.

3.2.3. remove_usb_printer.sh

If the printer is disconnected from USB, this script is triggered and executed with Udev rules and removes the printer from the CUPS server. This prevents unnecessary printer registrations in the system and keeps the printer database updated.

3.2.4. activate_airprint.sh

The activate_airprint.sh script was written to activate AirPrint functionality by creating a hotspot on the Raspberry Pi. This process allows older printers to be used wirelessly from modern devices. In the first step, the Raspberry Pi is configured as a Wi-Fi access point using the hostapd and dnsmasq tools. Initially, software that controls the Raspberry Pi's Wi-fi management, such as wpa_supplicant and NetworkManager, are disabled. Hostapd puts the Wi-Fi module into hotspot mode by setting the network's basic settings such as SSID and password. Meanwhile, dnsmasq assigns IP addresses to connected devices and enables communication between them.[7]

The script then starts the Avahi daemon for the printers that turn on the sharing feature on CUPS. Avahi allows printers to be automatically detected by other devices on the network via mDNS (Multicast DNS). This allows users to access the printer without manual IP addressing.

Finally, the script completes the integration of existing printers with CUPS. Printers on the CUPS server are queried and configured in accordance with the AirPrint protocol. Upon completion of the configurations, the CUPS service is restarted and the changes are applied. All of these processes allow printers to adapt to modern wireless printing needs.

3.2.5. deactivate_airprint.sh

The deactivate_airprint.sh script is used to return the device to its normal network configuration by disabling AirPrint functionality. First of all, the hostapd and dnsmasq services are stopped and the related settings are reset in order to disable the hotspot mode and make the device work in client mode again. Thus, the wpa_supplicant and NetworkManager services can be restarted. This step makes it possible for the device to reconnect to the default Wi-Fi network.

The Avahi daemon can be optionally stopped since it is the component that enables the publishing of printers over the AirPrint protocol. However, since the default is to keep the printers discoverable over the main Wi-Fi network, it is preferred to keep Avahi running. In this way, AirPrint is available even the hotspot mode is off.

Finally, the system reconnects to the existing Wi-Fi network, maintaining the device's network connectivity with full functionality. This configuration is set up to allow the device to switch between different modes quickly and seamlessly.

4. ENGINEERING STANDARDS and DESIGN CONSTRAINTS

Various engineering standards were used in this project to provide printer integration and wireless printing support. The standards used and their role in the project are described in detail below:

- CUPS and IPP
- Common UNIX Printing System (CUPS) implements the Internet Printing Protocol (IPP) standard and ensures cross-platform compatibility. This protocol makes it possible to integrate with Windows, Linux, macOS, iOS and Android systems. [8]
- Avahi and mDNS
- The Avahi daemon enables automatic discovery of printers via the mDNS (multicast DNS) protocol and offers compatibility with the AirPrint protocol. [9]
- Wi-Fi Compliance
- The wireless connectivity is designed in accordance with the IEEE 802.11 Wi-Fi standard and provides a reliable structure for access point creation and network sharing. [10]
- USB Standards
- The USB 2.0/3.0 standard was used to support reliable connection of printers. [11]
- Automation Scripts
- The automation scripts used in the project were developed in accordance with POSIX standards and provide Linux compatibility. [12]

The design constraints encountered in this project are categorized as follows:

Hardware Limitations:

- Raspberry Pi's Processing Capacity:
  The Raspberry Pi's limited processing power and memory led to performance limitations when multiple printers were connected at the same time.
- Number of USB Ports:
  Limited USB ports on the Raspberry Pi limited the number of printers that could be connected at the same time.

Driver and Printer Compatibility:

- Adapting Older Drivers to Modern Systems:
  Printer manufacturers have discontinued driver support for older devices, making it difficult to integrate these devices with modern systems. This process is further complicated by missing documentation or printer protocols that are incompatible with today's standards.
- Support and Access Difficulty from Printer Manufacturers:
  As some manufacturers do not offer updated driver or software support for older printer models, integration processes have often relied on solutions developed by open-source communities. However, such solutions do not offer broad enough support to cover all devices and require manual intervention.
- Lack of Technical Documentation:
  Limited technical documentation on the protocols and driver details of legacy printers is another factor that makes software optimizations difficult.

Economy:

- The low cost of the Raspberry Pi made the project a budget-friendly solution.

- Maintenance processes have been optimized to provide a sustainable structure with minimal cost.

Environment:

- Low power consumption makes the device an environmentally friendly solution.
- The reuse of old printers contributed to the reduction of electronic waste and environmental sustainability.

Manufacturability:

- All components used in the project were selected to be compatible with existing manufacturing technologies.
- The compact nature of the Raspberry Pi has enabled a portable and scalable solution.

Sustainability:

- The system is designed for durability and to support future upgrades.
- Reliability and flexibility are prioritized to ensure long-term user support.

5. RESULTS AND DISCUSSIONS

The project successfully accomplished the stated objectives and resulted in a number of important outcomes. The designed device successfully automated the integration of legacy printers into network environments and wireless printing support. Some limitations and challenges were also addressed, which should be considered in the future to achieve more effective results.

The device is operational without any additional configuration after only connecting to the power supply and establishing the printer USB connection. The tests showed that the system worked seamlessly with three different printer models and successfully added these devices to the CUPS server.

The system automatically performed the processes of detecting the printers, assigning the necessary drivers and providing wireless access with AirPrint support as intended. In particular, the zero-configuration capabilities of the Avahi daemon enabled the automatic discovery of printers on the network, which significantly improved the user experience. The low-cost and efficient infrastructure provided by the Raspberry Pi enabled this process to be carried out both quickly and reliably.

The outputs of the device provide information to the user by effectively reporting the status of the printers. The system starts by running the list_printer.sh script and first displays the “Script started” message with the date and time as shown in Figure 5. Then the message “waiting to be ready” is displayed, waiting for all services of the Raspberry Pi to be activated. This step is critical to ensure that all system services are started correctly. After the services have fully started, the “CUPS is active” message informs users that the CUPS service has been successfully activated. In the next step, the status of the printers is checked via CUPS and if there is a printer registered in the system, the brand and model of the printer is displayed as “Printer: Xerox Phaser 311” or similar message. However, if there is no printer registered in the system, the user is informed with the message “No printer found”. These outputs reveal that the device automates printer integration processes in a user-friendly way and provides information flow effectively.

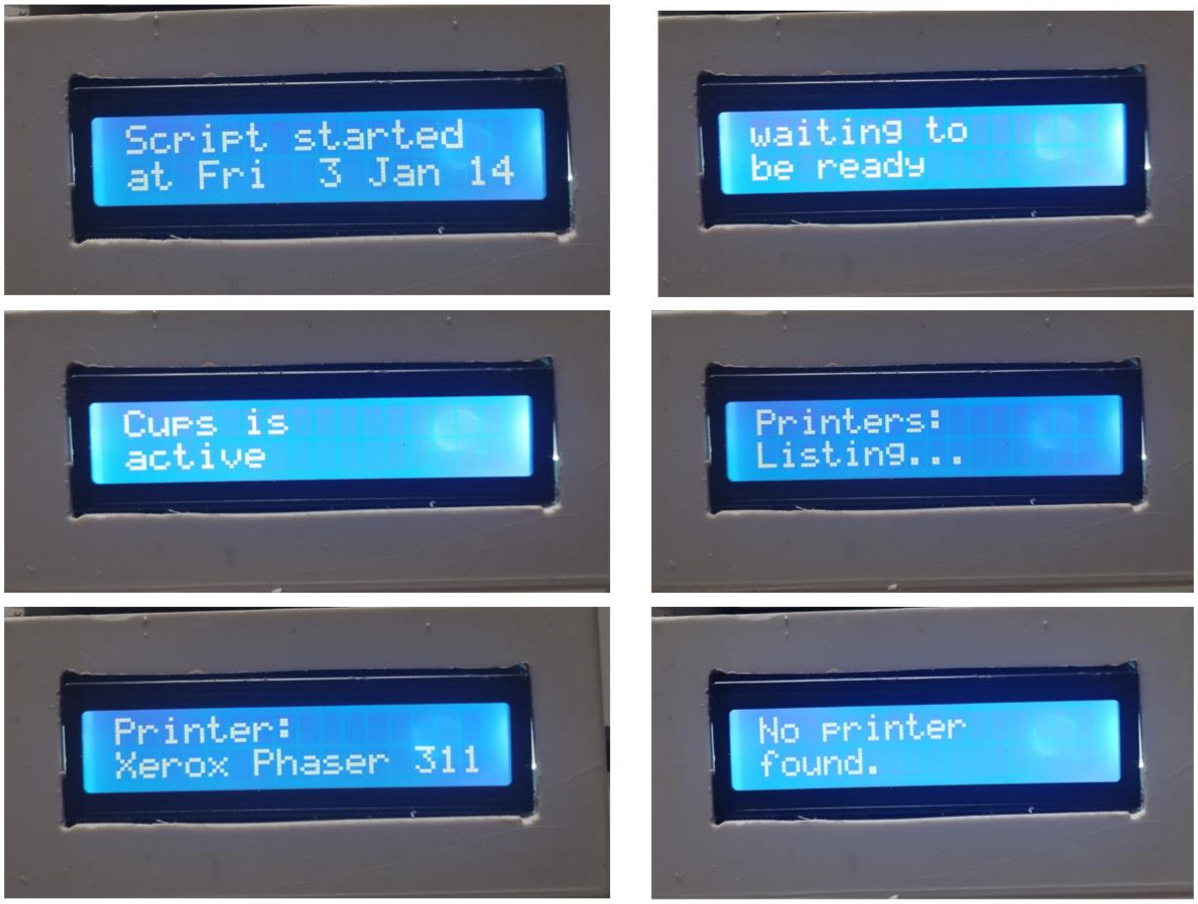

```
Script started
at Fri  3 Jan 14
```
```
waiting to
be ready
```
```
Cups is
active
```
```
Printers:
Listing...
```
```
Printer:
Xerox Phaser 311
```
```
No printer
found.
```

Figure 5: Output Screens of the list_printer.sh Script Demonstrating Printer Detection Process

The functionality of the add_usb_printer.sh script was effectively observed when the operating processes of the device continued to be internalized. The script runs automatically when the device is powered on and starts an integrated process. In the first step, the system checks if there is a USB connected printer. If the printer is not connected, the user is prompted to connect a printer (Figure 6). This step reflects the system's goal of minimizing user interaction.

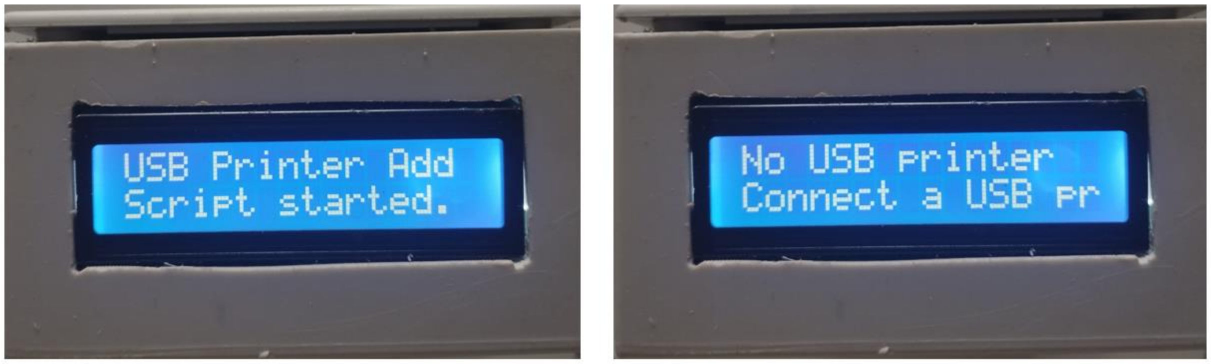
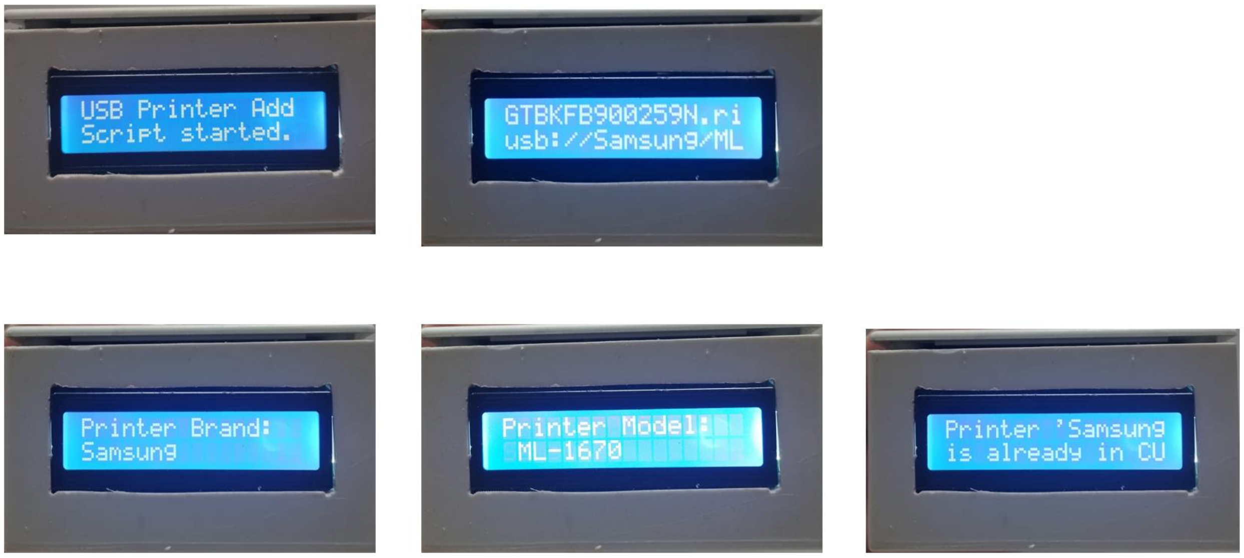

```
USB Printer Add
Script started.
```
```
No USB printer
Connect a USB pr
```

Figure 6: Output Screens of the add_usb_printer.sh Script in the Absence of a Connected Printer

Once a printer is connected via USB, the add_usb_printer.sh script is automatically triggered, and the URI of the printer is analyzed to determine the manufacturer and model information. If the device is already defined on the CUPS server, the user is shown the message “Already in CUPS” (Figure 7). However, if the device is not defined in CUPS, the system starts adding the printer, and the message “Adding new printer” is displayed on the screen. If the correct driver for the printer is not detected, the user is prompted to install the driver manually (Figure 8). If the proper driver has been found, the printer is successfully added to CUPS and the message that the process has been completed is displayed on the screen. In this process, all the steps for the script to run successfully are summarized in Figure 9. These processes effectively demonstrate the system's ability to integrate legacy printers into the network environment and its automation capacity.



```
USB Printer Add
Script started.
```

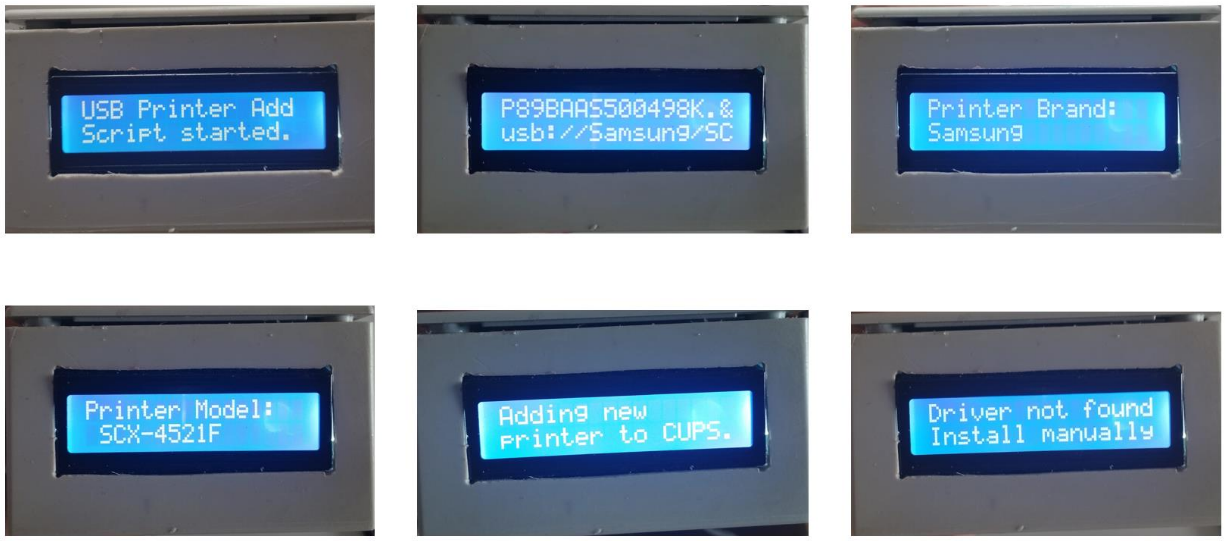
```
GTBKFB900259N.ri
usb://Samsung/ML
```

```
Printer Brand:
Samsung
```

```
Printer Model:
ML-1670
```

```
Printer 'Samsung
is already in CU
```

Figure 7: Sequential Output of the add_usb_printer.sh Script for Connected Printer Identification



```
USB Printer Add
Script started.
```

```
P89BAAS500498K.&
usb://Samsung/SC
```

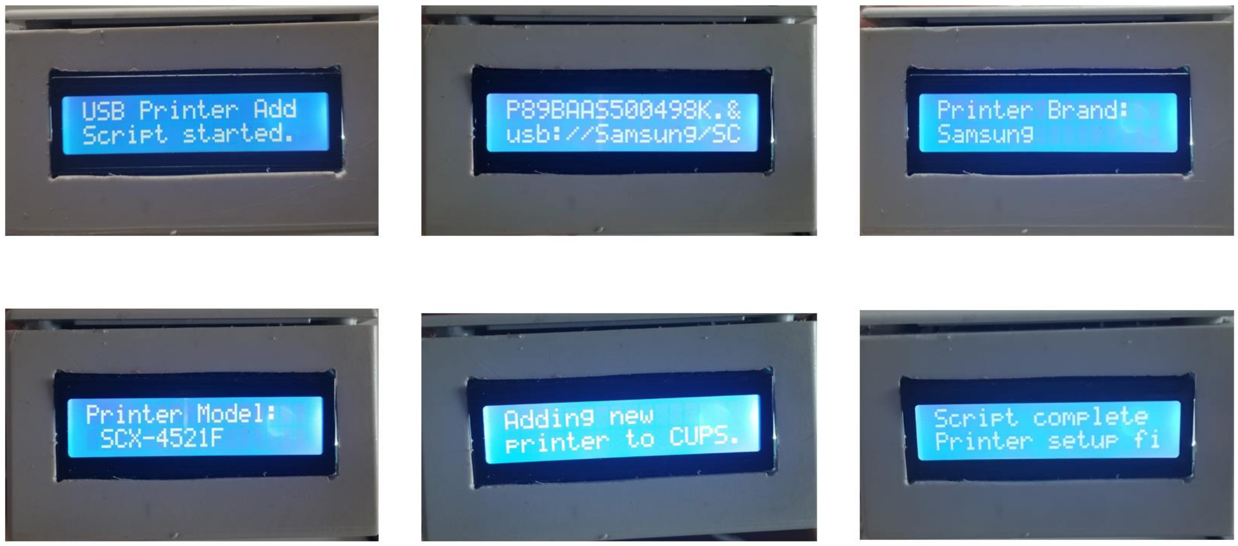
```
Printer Brand:
Samsung
```

```
Printer Model:
 SCX-4521F
```

```
Adding new
printer to CUPS.
```

```
Driver not found
Install manually
```

Figure 8: Output Screens of add_usb_printer.sh Script when no suitable driver is found by model name



```
USB Printer Add
Script started.
```

```
P89BAAS500498K.&
usb://Samsung/SC
```

```
Printer Brand:
Samsung
```

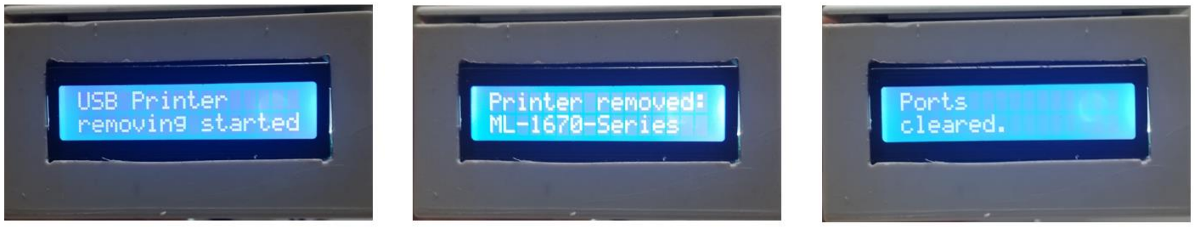
```
Printer Model:
 SCX-4521F
```

```
Adding new
printer to CUPS.
```

```
Script complete
Printer setup fi
```

Figure 9: Sequential Output of the add_usb_printer.sh Script when the new printer is successfully added

When the USB is disconnected, the device automatically triggers the remove_usb_printer.sh script and this is indicated on the screen with the message “USB Printer removing started”. The system then removes the printer from the CUPS server also, it shows the removed device on the screen as in Figure 10. This process is completed by clearing the ports associated with CUPS and the user receives the message “Ports cleared”. These steps enable the system to dynamically monitor the connection status of the printers and quickly remove the removed printers from the system.

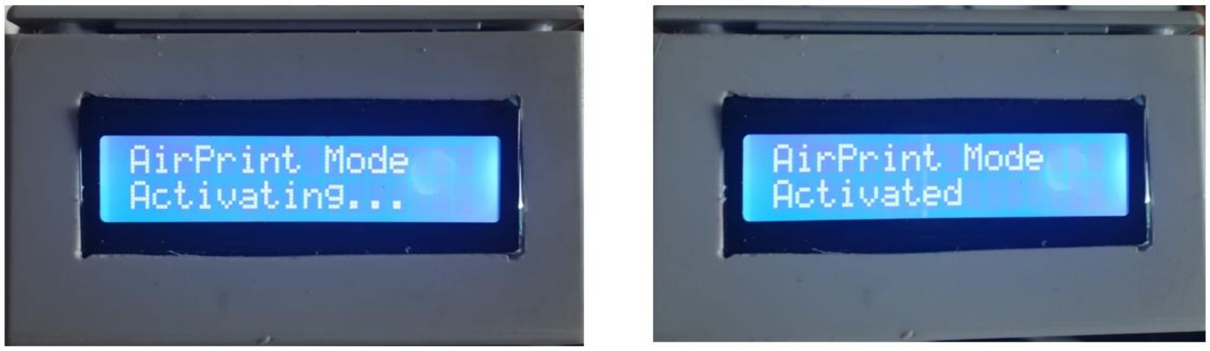

USB Printer
removing started

Printer removed:
ML-1670-Series

Ports
cleared.

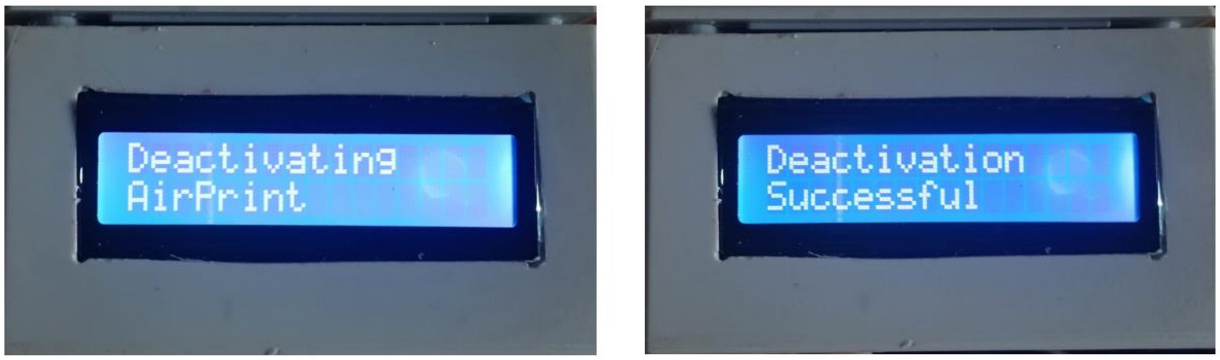
Figure 10: USB Printer Removal Process Displayed on the LCD Screen

The process of enabling and disabling AirPrint mode is shown in Figures 11 and 12. Figure 11 shows the process of activating AirPrint mode. After pressing the button, the message “AirPrint Mode Activating” is displayed during the activation of AirPrint mode, and the screen shows “AirPrint Mode Activated” when the process is completed. This process indicates that the system has successfully activated the wireless printing functionality. Similarly, Figure 12 shows the process of deactivating AirPrint mode. Deactivating AirPrint mode starts when the button is pressed a second time. First, the message “Deactivating AirPrint” is displayed and when the process is complete, the message “Deactivation Successful” appears on the screen.


AirPrint Mode
Activating...

AirPrint Mode
Activated

Figure 11: Screens Displayed During the Activation Process of AirPrint Mode


Deactivating
AirPrint

Deactivation
Successful

Figure 12: Screens Displayed During the Deactivation Process of AirPrint Mode

On the other hand, some technical problems and limitations emerged during the development process. The use of CUPS (Common UNIX Printing System) for printer integration requires printer drivers to be defined in this system. Since some printer drivers were not available in the CUPS driver repository, manual searches were required to find drivers for older printer models, which slowed down the integration process. To solve this problem, a wide range of driver support on the Raspberry Pi device was utilized. Open-source drivers such as Splix driver were installed, in addition to Foomatic database downloaded from GitHub to create a comprehensive driver collection. This approach made it possible to expand the range of printer models. However, the lack of support for some older printers by manufacturers and the discontinuation of their production processes make it difficult to develop a long-term solution for the integration of these devices. Over time, such devices are likely to lose compatibility with modern systems.

The naming inconsistencies that arise during the processing of model and manufacturer information in the URI (Uniform Resource Identifier) used during USB connection are another issue. In some cases, unexpected characters and different naming formats in the URI caused incompatibilities with CUPS. For example, when the Samsung ML-1675 printer was used for testing, it was identified as ML-1670_Series in the URI, but the CUPS driver for this printer was named Samsung_ML-1670.ppd. Such mismatches prevented the printers from being mapped to the correct drivers, requiring manual control corrections during the integration process. To address this issue, the commands used to extract model and manufacturer information from the URI have been optimized. However, despite these improvements, the first time a new printer is added to the system, it should be checked for naming mismatches. If necessary, manual optimization is required.

In conclusion, the system developed during this project has achieved a significant success by automating printer integration. The limitations and technical issues encountered highlighted the complexity of printer integration, but demonstrated that open-source tools can provide a solution. In the future, it is possible to make this system more inclusive with more advanced driver support and library additions of automation algorithms.

6. CONCLUSIONS

In summary, this research was able to successfully build a tool that facilitates wireless printing and automatically integrates legacy USB printers into contemporary network systems. Using CUPS (Common UNIX Printing System) as a core building component, the Raspberry Pi-based solution enabled self-management of printer detection and configuration. Also, it enabled wireless printing with the AirPrint protocol. The primary goal of the project is to reduce the dependency of users on technical assistance by replacing manual activities by a completely automated device.

The developed device worked seamlessly without any additional configuration, just by connecting it to a power supply and connecting a printer via USB. Tests on different printer models showed that the system successfully detected the printers, assigned the appropriate drivers and provided wireless access via the AirPrint protocol. Discoverability and print completion from smart devices significantly improved the user experience.

Another important impact of this project is the contribution of reusing old printers to reduce the amount of electronic waste. By enabling the use of old devices in modern systems, this solution is cost-effective and supports environmental sustainability. The successful integration of automation processes provided a user-friendly experience and eliminated wasted time.

However, various limitations were encountered during the project. CUPS' existing driver repository was insufficient for some older printer models, resulting in the need for manual driver search and integration. Although this problem was partially solved with the Foomatic database and third-party drivers, it was not possible to support all devices. In addition, naming inconsistencies when extracting model and manufacturer information from URI information necessitated manual intervention in some cases. These problems were largely resolved by optimizing the name extraction algorithms, but could not be completely eliminated.

This work can be improved in the future with advances such as increased driver support for a wider range of printers and the design of a dedicated processor to improve the cost and energy efficiency of the system. It can also be extended to a wider range of products by integrating more advanced algorithms and libraries tailored to each scenario for a completely autonomous and user-intervention-free configuration process.

In conclusion, this project presents an innovative approach to incorporate printers that are not adaptable to modern systems into contemporary network environments. Thus, it aimed to achieve an impact on user satisfaction and environmental sustainability. With its automation and user-friendliness, the device developed in the study has been successful in offering a new perspective for printer management.

REFERENCES

[1] IOGEAR. (n.d.). *Universal Ethernet to Wi-Fi Adapter - GWU637*. Retrieved January 14, 2025, from https://iogear.com/products/gwu637

[2] EasyTopTen. (n.d.). *Best Wireless Printer Adapters*. Retrieved January 14, 2025, from https://easytopten.com/best-wireless-printer-adapters/

[3] Google, "Google Cloud Print Overview." [Online]. Available: https://support.google.com

[4] Apple Inc. (n.d.). *Common UNIX Printing System (CUPS) Documentation*. Retrieved January 14, 2025, from https://www.cups.org/documentation.html

[5] The Avahi Project. (n.d.). *Avahi - Service Discovery for Linux using mDNS/DNS-SD*. Retrieved January 14, 2025, from https://www.avahi.org/

[6] S. K. Sood and S. K. Gupta, "Turning Wired Printers Wireless with Raspberry Pi," *International Journal of Computer Applications*, vol. 123, no. 11, pp. 1-5, Aug. 2015. [Online]. Available: https://www.academia.edu/36692754/Wireless_Printer_Fastener_using_Raspberry_Pi. [Accessed: Jan. 14, 2025].

[7] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Amazon Echo Dot or the Reverberating Secrets of IoT Devices," *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, Nov. 2021, pp. 1234-1245. [Online]. Available: https://dl.acm.org/doi/10.1145/3448300.3467820. [Accessed: Jan. 14, 2025].

[8] R. deBry, P. Hastings, T. Isaacson, and T. Powell, "Internet Printing Protocol/1.1: Model and Semantics," RFC 8011, Jan. 2017. [Online]. Available: https://www.ietf.org/rfc/rfc8011.txt

[9] Apple Inc., "Bonjour Protocol Specification," 2020. [Online]. Available: https://developer.apple.com/bonjour/

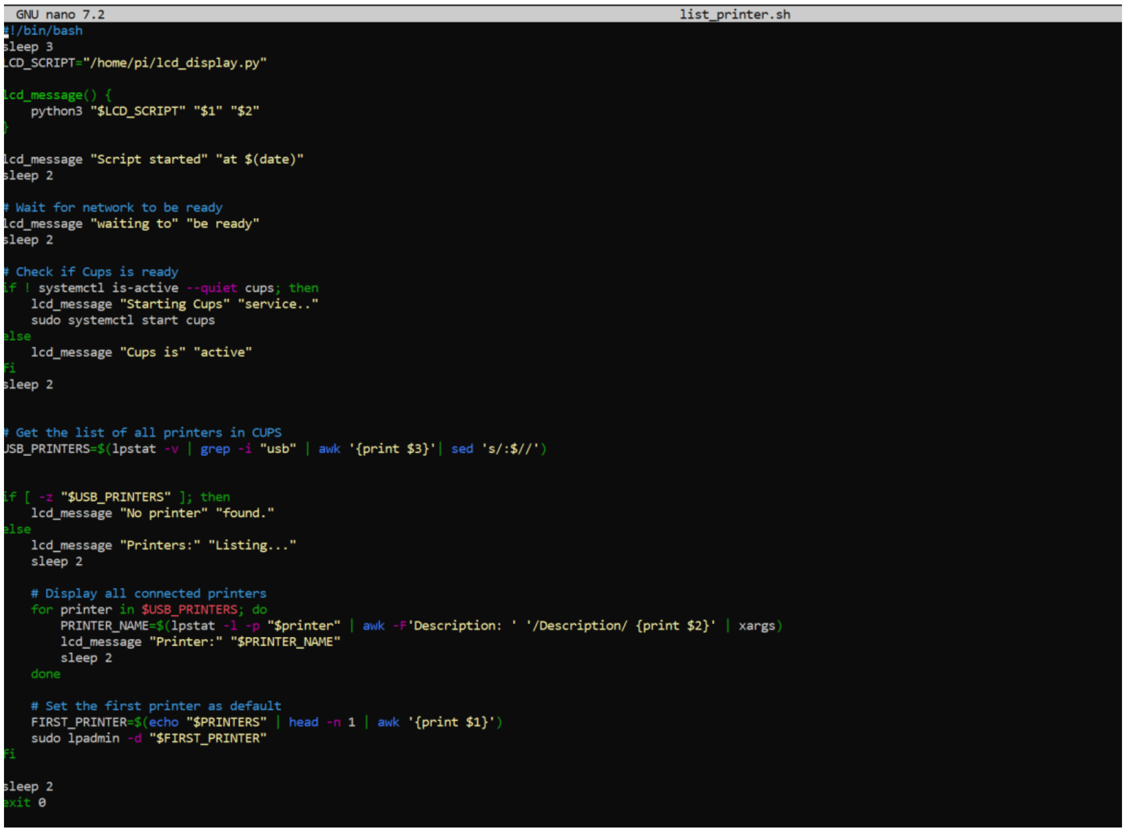
[10] IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements, IEEE Std 802.11, 2020. [Online]. Available: https://standards.ieee.org/

[11] USB Implementers Forum, "Universal Serial Bus Specification Revision 2.0," Apr. 2000. [Online]. Available: https://www.usb.org/documents

[12] IEEE Std 1003.1, "Standard for Information Technology—Portable Operating System Interface (POSIX)," 2017. [Online]. Available: https://standards.ieee.org/

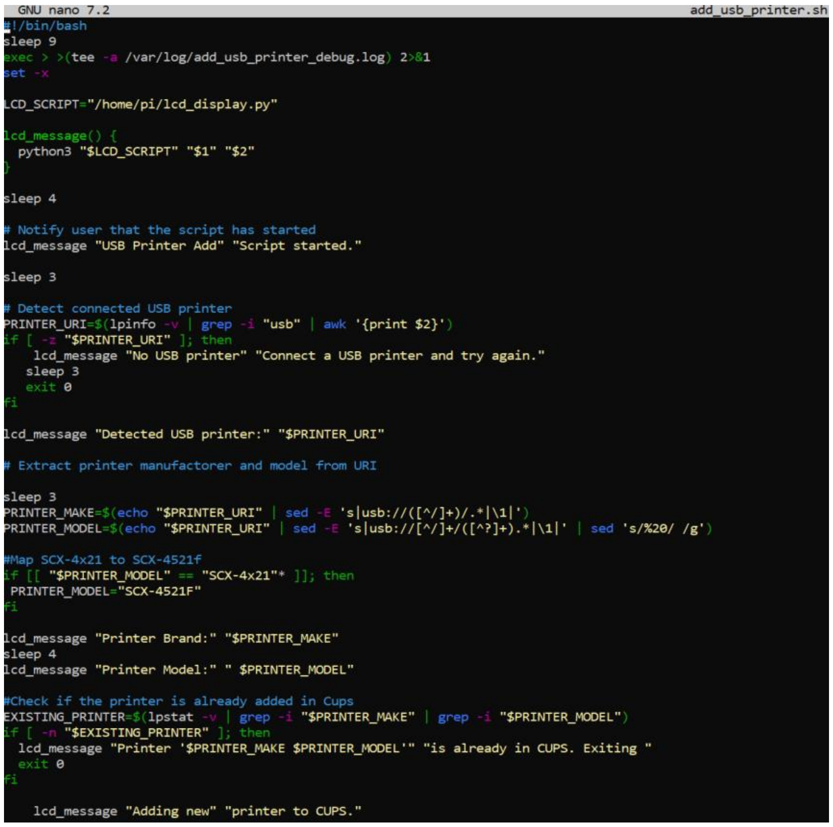
APPENDIX

The following section includes all the script files written and utilized during the project, providing detailed implementations and configurations for the system:



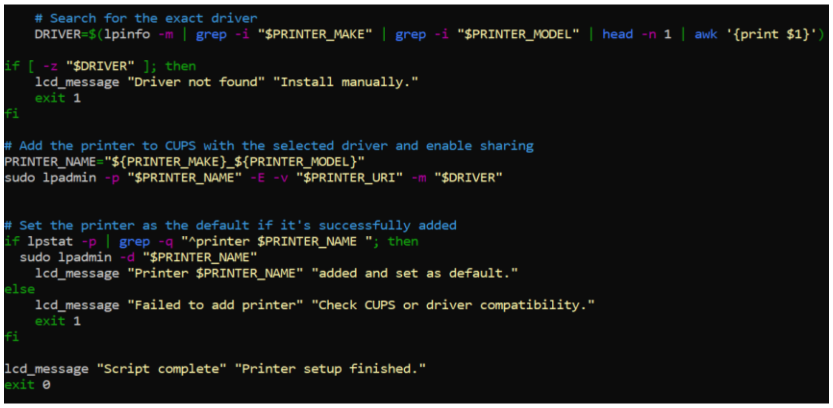
```
 GNU nano 7.2                                                                                    list_printer.sh
#!/bin/bash
sleep 3
LCD_SCRIPT="/home/pi/lcd_display.py"

lcd_message() {
    python3 "$LCD_SCRIPT" "$1" "$2"
}

lcd_message "Script started" "at $(date)"
sleep 2

# Wait for network to be ready
lcd_message "waiting to" "be ready"
sleep 2

# Check if Cups is ready
if ! systemctl is-active --quiet cups; then
    lcd_message "Starting Cups" "service.."
    sudo systemctl start cups
else
    lcd_message "Cups is" "active"
fi
sleep 2


# Get the list of all printers in CUPS
USB_PRINTERS=$(lpstat -v | grep -i "usb" | awk '{print $3}'| sed 's/:$//')


if [ -z "$USB_PRINTERS" ]; then
    lcd_message "No printer" "found."
else
    lcd_message "Printers:" "Listing..."
    sleep 2

    # Display all connected printers
    for printer in $USB_PRINTERS; do
        PRINTER_NAME=$(lpstat -l -p "$printer" | awk -F'Description: ' '/Description/ {print $2}' | xargs)
        lcd_message "Printer:" "$PRINTER_NAME"
        sleep 2
    done

    # Set the first printer as default
    FIRST_PRINTER=$(echo "$PRINTERS" | head -n 1 | awk '{print $1}')
    sudo lpadmin -d "$FIRST_PRINTER"
fi

sleep 2
exit 0
```

Figure 13: list_printer.sh script code



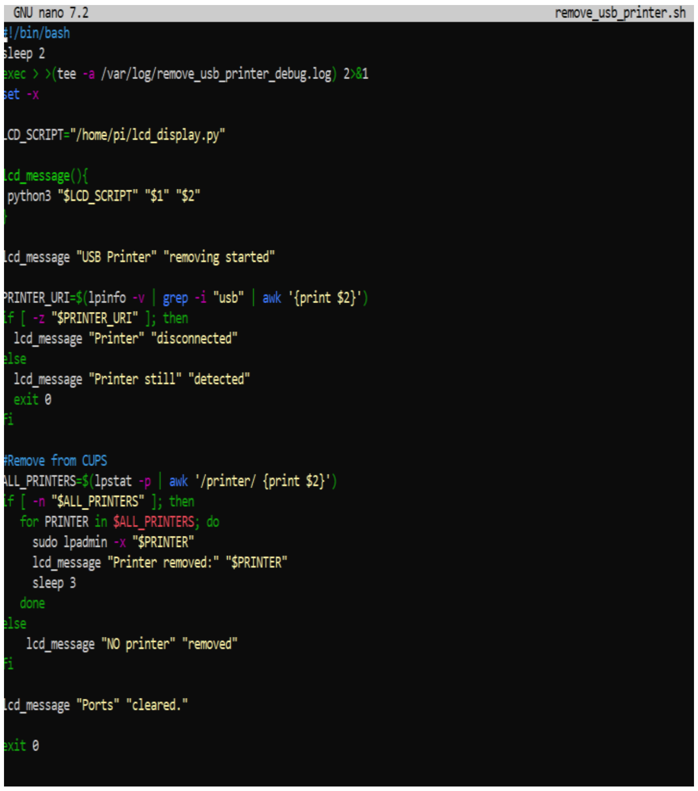
```
GNU nano 7.2                                                              add_usb_printer.sh
#!/bin/bash
sleep 9
exec > >(tee -a /var/log/add_usb_printer_debug.log) 2>&1
set -x

LCD_SCRIPT="/home/pi/lcd_display.py"

lcd_message() {
  python3 "$LCD_SCRIPT" "$1" "$2"
}

sleep 4

# Notify user that the script has started
lcd_message "USB Printer Add" "Script started."

sleep 3

# Detect connected USB printer
PRINTER_URI=$(lpinfo -v | grep -i "usb" | awk '{print $2}')
if [ -z "$PRINTER_URI" ]; then
    lcd_message "No USB printer" "Connect a USB printer and try again."
   sleep 3
   exit 0
fi

lcd_message "Detected USB printer:" "$PRINTER_URI"

# Extract printer manufactorer and model from URI

sleep 3
PRINTER_MAKE=$(echo "$PRINTER_URI" | sed -E 's|usb://([^/]+)/.*|\1|')
PRINTER_MODEL=$(echo "$PRINTER_URI" | sed -E 's|usb://[^/]+/([^?]+).*|\1|' | sed 's/%20/ /g')

#Map SCX-4x21 to SCX-4521f
if [[ "$PRINTER_MODEL" == "SCX-4x21"* ]]; then
 PRINTER_MODEL="SCX-4521F"
fi

lcd_message "Printer Brand:" "$PRINTER_MAKE"
sleep 4
lcd_message "Printer Model:" " $PRINTER_MODEL"

#Check if the printer is already added in Cups
EXISTING_PRINTER=$(lpstat -v | grep -i "$PRINTER_MAKE" | grep -i "$PRINTER_MODEL")
if [ -n "$EXISTING_PRINTER" ]; then
  lcd_message "Printer '$PRINTER_MAKE $PRINTER_MODEL'" "is already in CUPS. Exiting "
  exit 0
fi

    lcd_message "Adding new" "printer to CUPS."
```

Figure 14: add_usb_printer.sh script code



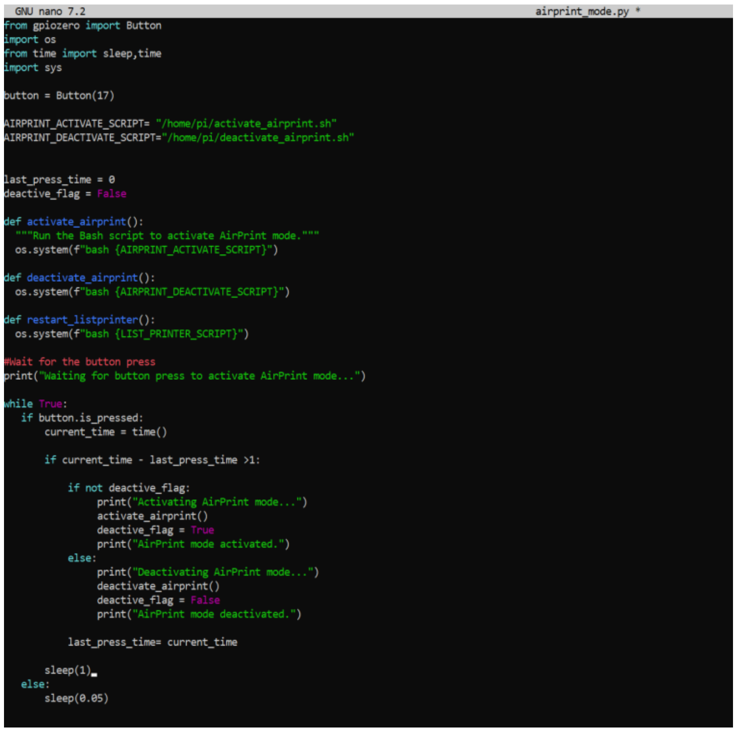
```
    # Search for the exact driver
    DRIVER=$(lpinfo -m | grep -i "$PRINTER_MAKE" | grep -i "$PRINTER_MODEL" | head -n 1 | awk '{print $1}')

if [ -z "$DRIVER" ]; then
    lcd_message "Driver not found" "Install manually."
    exit 1
fi

# Add the printer to CUPS with the selected driver and enable sharing
PRINTER_NAME="${PRINTER_MAKE}_${PRINTER_MODEL}"
sudo lpadmin -p "$PRINTER_NAME" -E -v "$PRINTER_URI" -m "$DRIVER"


# Set the printer as the default if it's successfully added
if lpstat -p | grep -q "^printer $PRINTER_NAME "; then
  sudo lpadmin -d "$PRINTER_NAME"
    lcd_message "Printer $PRINTER_NAME" "added and set as default."
else
    lcd_message "Failed to add printer" "Check CUPS or driver compatibility."
    exit 1
fi

lcd_message "Script complete" "Printer setup finished."
exit 0
```

Figure 15: add_usb_printer.sh script code continued



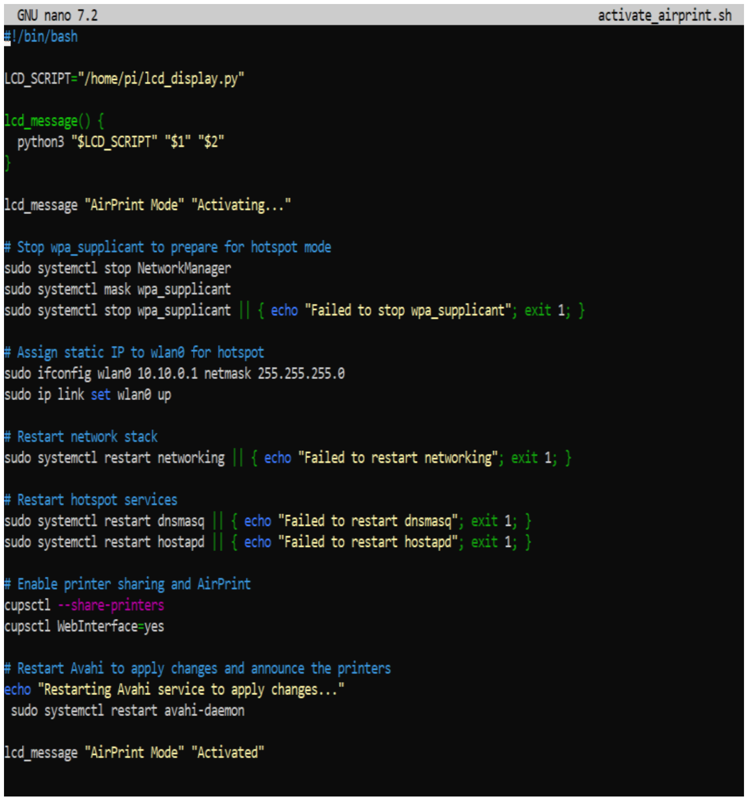
```
GNU nano 7.2                                                    remove_usb_printer.sh
#!/bin/bash
sleep 2
exec > >(tee -a /var/log/remove_usb_printer_debug.log) 2>&1
set -x

LCD_SCRIPT="/home/pi/lcd_display.py"

lcd_message(){
 python3 "$LCD_SCRIPT" "$1" "$2"
}

lcd_message "USB Printer" "removing started"

PRINTER_URI=$(lpinfo -v | grep -i "usb" | awk '{print $2}')
if [ -z "$PRINTER_URI" ]; then
  lcd_message "Printer" "disconnected"
else
  lcd_message "Printer still" "detected"
  exit 0
fi

#Remove from CUPS
ALL_PRINTERS=$(lpstat -p | awk '/printer/ {print $2}')
if [ -n "$ALL_PRINTERS" ]; then
   for PRINTER in $ALL_PRINTERS; do
     sudo lpadmin -x "$PRINTER"
     lcd_message "Printer removed:" "$PRINTER"
     sleep 3
   done
else
    lcd_message "NO printer" "removed"
fi

lcd_message "Ports" "cleared."

exit 0
```

Figure 16: remove_usb_printer.sh script code



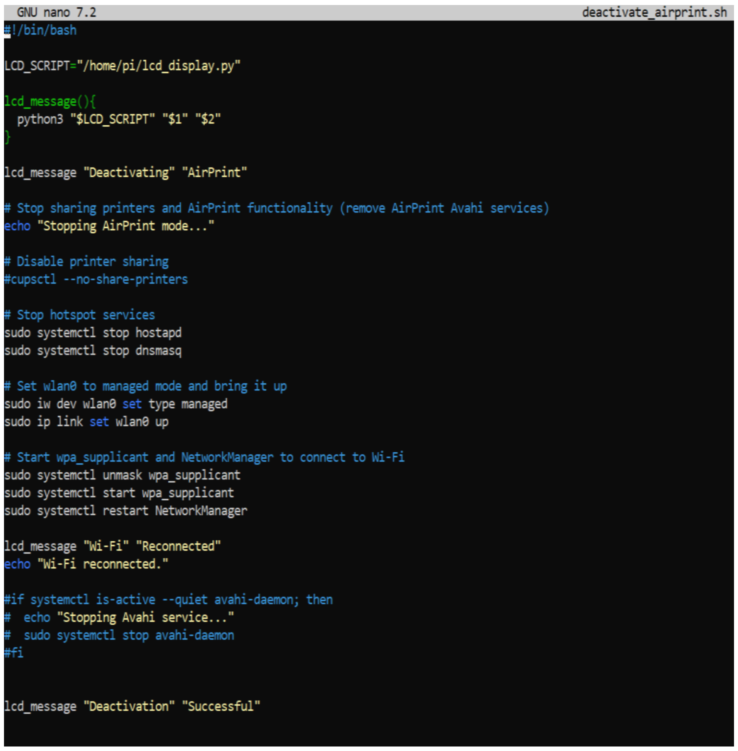
```
GNU nano 7.2                                                    airprint_mode.py *
from gpiozero import Button
import os
from time import sleep,time
import sys

button = Button(17)

AIRPRINT_ACTIVATE_SCRIPT= "/home/pi/activate_airprint.sh"
AIRPRINT_DEACTIVATE_SCRIPT="/home/pi/deactivate_airprint.sh"


last_press_time = 0
deactive_flag = False

def activate_airprint():
  """Run the Bash script to activate AirPrint mode."""
  os.system(f"bash {AIRPRINT_ACTIVATE_SCRIPT}")

def deactivate_airprint():
  os.system(f"bash {AIRPRINT_DEACTIVATE_SCRIPT}")

def restart_listprinter():
  os.system(f"bash {LIST_PRINTER_SCRIPT}")

#Wait for the button press
print("Waiting for button press to activate AirPrint mode...")

while True:
   if button.is_pressed:
       current_time = time()

       if current_time - last_press_time >1:

           if not deactive_flag:
                print("Activating AirPrint mode...")
                activate_airprint()
                deactive_flag = True
                print("AirPrint mode activated.")
           else:
                print("Deactivating AirPrint mode...")
                deactivate_airprint()
                deactive_flag = False
                print("AirPrint mode deactivated.")

           last_press_time= current_time

       sleep(1)
   else:
       sleep(0.05)
```

Figure 17: airprint_mode.py script code



```
GNU nano 7.2                                                    activate_airprint.sh
#!/bin/bash

LCD_SCRIPT="/home/pi/lcd_display.py"

lcd_message() {
  python3 "$LCD_SCRIPT" "$1" "$2"
}

lcd_message "AirPrint Mode" "Activating..."

# Stop wpa_supplicant to prepare for hotspot mode
sudo systemctl stop NetworkManager
sudo systemctl mask wpa_supplicant
sudo systemctl stop wpa_supplicant || { echo "Failed to stop wpa_supplicant"; exit 1; }

# Assign static IP to wlan0 for hotspot
sudo ifconfig wlan0 10.10.0.1 netmask 255.255.255.0
sudo ip link set wlan0 up

# Restart network stack
sudo systemctl restart networking || { echo "Failed to restart networking"; exit 1; }

# Restart hotspot services
sudo systemctl restart dnsmasq || { echo "Failed to restart dnsmasq"; exit 1; }
sudo systemctl restart hostapd || { echo "Failed to restart hostapd"; exit 1; }

# Enable printer sharing and AirPrint
cupsctl --share-printers
cupsctl WebInterface=yes

# Restart Avahi to apply changes and announce the printers
echo "Restarting Avahi service to apply changes..."
 sudo systemctl restart avahi-daemon

lcd_message "AirPrint Mode" "Activated"
```

Figure 18: activate_airprint.sh script code



```
GNU nano 7.2                                                  deactivate_airprint.sh
#!/bin/bash

LCD_SCRIPT="/home/pi/lcd_display.py"

lcd_message(){
  python3 "$LCD_SCRIPT" "$1" "$2"
}

lcd_message "Deactivating" "AirPrint"

# Stop sharing printers and AirPrint functionality (remove AirPrint Avahi services)
echo "Stopping AirPrint mode..."

# Disable printer sharing
#cupsctl --no-share-printers

# Stop hotspot services
sudo systemctl stop hostapd
sudo systemctl stop dnsmasq

# Set wlan0 to managed mode and bring it up
sudo iw dev wlan0 set type managed
sudo ip link set wlan0 up

# Start wpa_supplicant and NetworkManager to connect to Wi-Fi
sudo systemctl unmask wpa_supplicant
sudo systemctl start wpa_supplicant
sudo systemctl restart NetworkManager

lcd_message "Wi-Fi" "Reconnected"
echo "Wi-Fi reconnected."

#if systemctl is-active --quiet avahi-daemon; then
#  echo "Stopping Avahi service..."
#  sudo systemctl stop avahi-daemon
#fi


lcd_message "Deactivation" "Successful"
```

Figure 19: deactivate_airprint.sh script code